This article examines the complexity of resource-allocation decisions for resource-constrained project scheduling. The allocation decisions are modeled by means of precedence networks and are closely related to ES-policies, which are a type of scheduling policies introduced for stochastic scheduling. We find that even a number of ‘surrogate’ objective functions, whose use has recently been proposed by multiple sources, lead to hard problems. Our results confirm that resource allocation is difficult even when objective-function evaluation by itself is not intractable.

Keywords: project scheduling, resource constraints, resource allocation, complexity.

1 Introduction

A project can be informally defined for our purposes as a set of precedence-related activities that have to be performed using diverse and usually limited resources. Project scheduling is the part of project management that deals with deciding when to execute which activities and how to allocate resources to the activities. In practice, virtually all project managers are confronted with resource scarceness. In such cases, the Resource-Constrained Project-Scheduling Problem (RCPSP) arises.

During the execution of a project, unexpected events can occur that cause deviations from the initially projected schedule. Some of the most frequently encountered sources of schedule disruption are equipment failure, bad-weather delay, worker absenteeism, under-estimation or over-estimation of an activity’s work content, . . . The majority of these sources of uncertainty can be modeled as an activity duration increase or decrease. In a context of significant duration uncertainty, the study domain of stochastic scheduling proposes not to build a complete schedule before the beginning of the project, because it can often not be followed anyway, but rather to simply start new
activities gradually as resources become available, and thus stepwise build a
schedule as the project’s execution progresses.

The objective function in stochastic scheduling is usually the expected
value of a function of the activity durations, most frequently the project
makespan. It turns out that exact determination of these objectives for
a given solution is mostly intractable. For this reason, numerous recent
sources resort to optimizing so-called ‘surrogate’ quality measures, which are
alternative objective functions that can be efficiently evaluated and for which
it is hoped that solutions that perform well for these surrogate measures will
also be of high quality when evaluated on the actual objectives. Leon et
al. [23], for instance, state: “Our approach is to develop surrogate measures
that are strongly correlated with expected delay and makespan. Although
this approach will result in an approximation . . . , the measures are easy to
compute and are applicable to a variety of cases present in real situations”.

In this article, we examine the complexity of resource-allocation deci-
sions for resource-constrained project scheduling. The allocation decisions
are modeled by means of precedence networks and are closely related to
ES-policies, which are a type of scheduling policies introduced for stochas-
tic scheduling. Our results confirm that resource allocation is difficult even
when objective-function evaluation by itself is not intractable. This article
is a continuation of the work in [26]; a summary of the relevant concepts
and definitions is provided in Section 2. We give a detailed statement of
the problems examined, together with links to the related literature, in Sec-
tion 3. Our main results are described in Sections 4, 5 and 6, which treat
resource allocation without initial schedule, with initial schedule, and joint
scheduling and resource allocation, respectively. We summarize our findings
in Section 7.

2 Definitions and previous results

The project scheduling problem that we examine is discussed in [9, 21, 26,
31, 32, 38] and consists in the non-preemptive scheduling of a set \( N = \{0, 1, \ldots, n\} \) of activities subject to a precedence relation \( A \subset N \times N \) on
a set \( K \) of renewable resource types with availability \( a_k \) for type \( k \in K \),
where a precedence relation \( E \) is a a binary relation \( E \subset N \times N \) that is
irreflexive and transitive (i.e., a strict partial order on \( N \)). Each activity \( i \)
has a duration \( d_i \in \mathbb{N} \) and occupies \( r_{ik} \in \mathbb{N} \) units of resource type \( k \) during
its execution. The (dummy) activities 0 and \( n \) have zero duration and zero
resource usage; we assume that \( d_i > 0 \) for \( i \neq 0, n \). The value \( s_i \geq 0 \) is the
starting time of activity \( i \); we call the vector \( s = (s_0, \ldots, s_n) \) a schedule. The
RCPSP $\Gamma(N, A, d, r, a)$ aims at finding a feasible schedule $s$ that minimizes a cost function, usually $s_n$ (see, for example, [10, 34]), where the vectors $d$, $r$ and $a$ collect activity durations, resource requirements and resource availabilities, respectively. The precedence constraints require $s_i + d_i \leq s_j$ for all $(i, j) \in A$, while the resource constraints impose that no more than $a_k$ resource units of each resource type can concurrently be used.

Following the line of [26], we represent resource-allocation decisions by means of a resource flow $f$, which assigns a value $f(i, j, k) \in \mathbb{N}$ to each triple $(i, j, k) \in N \times N \times K$, equal to the number of resource units of type $k$ that are transferred from the end of activity $i$ to the start of $j$. For each activity $i \in N \setminus \{0,n\}$ and each resource type $k$, a resource flow $f$ satisfies $\sum_{j \in N} f(i, j, k) = \sum_{j \in N} f(i, j, k) = r_{ik}$. The resources are sent into the network from the dummy start node 0 and received at the end node $n$: $\sum_{j \in N} f(0, j, k) = \sum_{j \in N} f(j, n, k) = a_k, \forall k \in K$. The activity pairs that are used by $f$ are gathered into set $\phi(f) = \{(i, j) \in N \times N : (\exists k \in K|f(i, j, k) > 0)\}$; the arcs that do not coincide with technological precedence constraints are collected in set $C(f) = \phi(f) \setminus A$. We say that a resource flow $f$ is feasible when $G(N, A \cup C(f))$ is acyclic.

For given values $r_{ik}$, a set of activities $F \subseteq N$ is a forbidden set of a precedence relation $E$ if it is an anti-chain of $E$ and if $\sum_{i \in F} r_{ik} > a_k$ for at least one $k \in K$. A subset-minimal forbidden set is called a minimal forbidden set or MFS. The set of all MFSs for a precedence relation $E$ is $\mathcal{F}(E)$. For a precedence relation $E$, we let $T(E)$ denote its transitive closure. A set of activity pairs $X \subseteq (N \times N) \setminus A$ is called a sufficient set or selection if $\mathcal{F}(T(A \cup X)) = \emptyset$ and $G(N, A \cup X)$ is acyclic. Put differently, the set $X$ is chosen such that we can ignore the resource constraints if we respect the extended set of precedence constraints $A \cup X$. The Main Representation Theorem of Bartusch et al. [5] states that the RCPSP with a regular objective function reduces to the search for an appropriate sufficient set. In stochastic scheduling, an ES-policy (short for early-start policy) is a valid approach to project execution: an ES-policy is parameterized by a sufficient selection $X$ and determines activity starting times by a simple early-start critical-path recursion in the graph $G(N, A \cup X)$. In this text, we sometimes refer to a sufficient selection as an ES-selection.

For a feasible resource flow $f$, the set $X = C(f)$ is sufficient, which follows from the fact that the summed resource usage of an anti-chain of $T(A \cup C(f))$ never exceeds $a_k$ for any $k \in K$. It can also be shown that if $X$ is a sufficient set then a feasible flow $f$ exists with $A \cup C(f) \subseteq T(A \cup X)$. The transitive reduction $t(E)$ of a binary relation $E$ on a set $N$ is the minimal relation on $N$ such that $T(t(E)) = T(E)$. We call an ES-selection $X$ dominant if $(T(A \cup X) \setminus \{(i, j)\}) \setminus A$ is not sufficient for all $(i, j) \in t(A \cup X) \setminus A$. A
selection that is not dominant imposes stricter precedence constraints than necessary.

In the order-theoretic approach to scheduling [31, 32], a search for an optimal schedule is replaced by the search for an optimal partial order. The following theorem is one of the main results of [26]:

**Theorem 1.** For any regular objective function, using the order-theoretic approach to scheduling, there exists an optimal solution that is a subset-minimal dominant ES-selection.

### 3 The search for optimal policies

The objective function in stochastic scheduling is usually the expected value of a function of the activity durations, most frequently the project makespan. The evaluation of the expected makespan of an ES-policy requires knowledge of the expected critical-path length of the project with an extended precedence network. It turns out that exact determination of such objectives is usually unrealistic [14, 30] in that it is highly unlikely that it could be done in polynomial time, and simulation is normally used, cf. [20, 32, 38]. According to Adlakha and Kulkarni [1], the difficulty arises from two sources: (1) the number of paths can grow exponentially in the number of activities; (2) even when the activity durations are independent, the path lengths are generally dependent, as there are activities common to more than one path.

This paper examines the search for desirable ES-policies (‘desirable’ meaning optimal for a specific objective function, within a specific class). In particular, we evaluate the difficulty of resource allocation when objective-function evaluation by itself is not intractable, which is the case for a number of recently proposed surrogate quality measures. We discuss the complexity status of the corresponding optimization problems (in Section 4) and we also investigate the special case of resource allocation for an input schedule (Section 5) as well as the combined problem of resource allocation and scheduling (Section 6).

Policella [35] proposes to optimize simple functions of the constructed graph’s parameters and thereby implicitly circumvents the intractability referred to supra. A first such function is borrowed from Aloulou and Portmann [3] and is proportional to the number of incomparable activities in $T(A \cup X)$. This objective is in line with suggestions by Hazir [16], Hazir et al. [17] and Chhourou and Haouari [7], who minimize the sum over the activities of the number of ‘direct and indirect’ successors. Deblaere et al. [9] pursue the minimization of the number of flow-carrying arcs, although their motivation is more in line with minimization of the size of the sufficient arc set. This
latter objective is also implicitly considered in Hazir [16] and Hazir et al. [17], where the effect of minimizing the total number of direct successors is examined. Nassar and Hegab [33] propose to minimize only the ‘direct’ arcs and not to focus on what they call ‘redundant’ arcs (arcs not in \( t(A \cup X) \)). Liu et al. [28] retain ‘the number of added disjunctive arcs’ as one of their quality measures.

From these cited sources, we distill the following three related optimization problems:

Problem MinSuff  
 Instance: RCPSP-instance \( \Gamma(N, A, d, r, a) \).
 Goal: Find a sufficient ES-selection \( X \subset N \times N \) with minimum cardinality.

Problem MinFlowArcs  
 Instance: RCPSP-instance \( \Gamma(N, A, d, r, a) \).
 Goal: Find a feasible resource flow \( f \) in \( G(N, N \times N) \) that minimizes \( |C(f)| \).

Problem MaxIncomp  
 Instance: RCPSP-instance \( \Gamma(N, A, d, r, a) \).
 Goal: Find a sufficient ES-selection \( X \subset N \times N \) that minimizes \( |T(A \cup X)| \).

MinSuff is most interesting from the part of minimal amendment: it yields the lowest number of extra arcs to add to the network. MinFlowArcs best reflects the need for limitation of inter-job communication overhead. Finally, MaxIncomp is useful especially when activity durations are variable: each pair of activities that becomes comparable gives rise to potential starting-time disruptions.

It can be shown that problem MaxIncomp is equivalent to finding a resource flow \( f^* = \arg \min_f |T(A \cup C(f))| \). We define an activity pair \((i, j) \in C(f)\) to be minimal with respect to a resource flow \( f \) if \((i, j) \in t(A \cup C(f))\), so if apart from arc \((i, j)\), no path \( i \rightarrow j \) exists in graph \( G(N, A \cup C(f)) \); note that a minimal arc is not in \( A \). The problem MinSuff is equivalent to finding a feasible flow with a minimum number of minimal arcs.

The following theorem holds:

**Theorem 2.** For MaxIncomp, there exists an optimal solution that is a subset-minimal dominant ES-selection.

**Proof:** Similar to the proof of Theorem 1. \(\square\)

For MinSuff and MinFlowArcs, on the other hand, a result similar to Theorem 2 cannot be obtained, which will be illustrated below. If an optimization
problem is devised to produce a heuristic solution to a more difficult problem (e.g., to the stochastic RCPSP), this is best kept in mind: Theorem 1 underlined the importance of minimal dominant selections for regular objective functions! The examples below illustrate that for some of the surrogate measures studied, all optimal solutions can be dominated for all regular objectives, which emphasizes the importance of a well-considered choice for such alternative quality measures.

Consider an example for which the initial precedence graph is visualized in Figure 1(a). With one resource type ($|K| = 1$), let $a_1 = r_1 = 2$, $r_3 = r_4 = 1$ and $r_2 = 0$. The MFSs are $\mathcal{F}(A) = \{\{1,3\}, \{1,4\}\}$, so $T(A \cup X)$ will need to contain at least two arcs more than $A$ for a sufficient ES-selection $X$. If we choose $X^* = \{(1,2)\}$ then $|X^*| = 1$, which is the unique optimal solution for MinSuff. According to our definitions, however, $X^*$ is not dominant. This is logical because $\{1,2\}$ is not even a part of any MFS (cf. Lemma 4 in [26]).

A feasible resource flow $f$ is called *dominant* if $f$ has no minimal activity pair $(i,j)$ such that a feasible flow $f^*$ exists with $C(f^*) \subset T(A \cup C(f)) \setminus \{(i,j)\}$. In a similar way as our foregoing observations for MinSuff, dominant flows are not always optimal for MinFlowArcs. We consider a project with precedence graph $G(N, A)$ as depicted in Figure 1(b). For $|K| = 1$ (one resource type), let $a_1 = 4$, $r_1 = r_2 = 2$ and $r_3 = r_4 = r_5 = r_6 = 1$; $T(A \cup X)$ will again need to contain at least two extra arcs for any sufficient selection $X$. Multiple feasible flows $f^*$ exist with $C(f^*) = \{(1,2)\}$, which are the only optimal solutions for MinFlowArcs. These flows $f^*$ are not dominant, however. The link between dominant resource flows and dominant ES-selections is discussed in [26].

Figure 1: Precedence constraints for the counterexamples; dummy start and end nodes are not included.
4 The complexity of resource allocation

We now turn our attention to the complexity status of the optimization problems at hand. We use the following decision problem:

Problem X3C (Exact cover by 3-sets)
Instance: A finite set $Q$ with $|Q| = 3q$ and a collection $S$ of 3-element subsets of $Q$.
Question: Does $S$ contain an exact cover for $Q$, that is, a subcollection $S' \subseteq S$ such that every element of $Q$ occurs in exactly one member of $S'$?

We know that X3C is NP-complete (see e.g. [12], problem SP2). Without loss of generality, we assume that $|S| \geq q$.

Theorem 3. Problem MinFlowArcs is NP-hard.

Proof: For an arbitrary instance of X3C, we construct an RCPSP-instance as follows. $N = \{0, t\} \cup Q \cup S$, where 0 and $t$ are dummy start and end, respectively (and thereby also predecessor, respectively successor, of all other activities). For all $c \in S, u \in Q : (c, u) \in A \iff u \in c$; apart from these pairs and those relating to 0 and $t$, $A$ is empty. We work with one resource type ($|K| = 1$) with availability $a_1 = |Q|$, with resource usage $r_c = 3$ for each $c \in S$ and $r_u = 1$ for all $u \in Q$. Activity durations are irrelevant here.

Our focus is on the minimization of $|C(f)|$. A lower bound on the objective-function value for a feasible resource flow $f$ is $LB = |S| - q$, because each (non-dummy) activity either obtains all its resource units from 0, or from another predecessor in $G(N, A)$, or it adds at least one unit to the objective function. If we neglect $Q$ then we still have $|S|$ activities with only 0 as predecessor (namely, all activities in $S$). Since $a_1 = 3q$, activity 0 can supply all resource units for at most $q$ of those activities.

Define $\alpha = \{i \in N : f(0, i, 1) > 0\}$; a feasible flow $f$ has $|C(f)| = LB$ only if $\alpha \subseteq S, |\alpha| = q$ and and for all $i \in \alpha, f(0, i, 1) = 3$. Another necessary condition is that for all $c \in S \setminus \alpha : |\{i \in N : f(i, c, 1) > 0\}| = 1$, which means that for each $c \in S \setminus \alpha$ there is an edge $(c_1, c) \in C(f)$ with $c_1 \in S$ and $f(c_1, c, 1) = 3$. This leads us to the conclusion that for any feasible flow $f$ with $|C(f)| = LB$, the set of arcs $Z = \{(i, j) \in T(A \cup C(f))|i \in \{0\} \cup S \land j \in Q \cup \{t\}\}$ is a cut in network $T(A \cup C(f))$ and $\{(i, j) \in T(A \cup C(f))|j \in \{0\} \cup S \land i \in Q \cup \{t\}\} = \emptyset$. Therefore, the sum of the flow values on the arcs in $Z$ is $3q$ for any feasible $f$.

If the answer to X3C is ‘yes’, it is easily shown that a feasible flow $f$ with $|C(f)| = LB$ exists.
Suppose that the answer to X3C is ‘no’. In this case, for each feasible flow $f$, all or part of the resource units used by at least $q + 1$ elements of $S$ are sent to the elements of $Q$, and/or there is resource flow between elements of $Q$, and/or $C(f)$ contains arcs $(c, u)$ with $c \in S$ and $u \in Q$. In the second and third case, it is immediate that $|C(f)| > LB$; we further investigate the case when there is no $u \in Q$ with $(i, u) \in C(f)$, so $\sum_{c \in S} f(c, u, 1) = r_u$ for each $u \in Q$. Since $\sum_{u \in Q} r_u = 3q$, there is no $c \in S$ with $f(c, t, 1) > 0$. Yet, all or part of the resource units used by at least $q + 1$ elements of $S$ are led to the elements of $Q$. Hence, there are at least two activities in $S$ that pass on either one or two resource units, but not three, to another activity in $S$. We noted above that this renders the equality $|C(f)| = LB$ impossible.

This concludes the proof: the answer to an arbitrary X3C-instance is ‘yes’ if and only if $|C(f)| = LB$ for the corresponding MinFlowArcs-instance, and the transformation can clearly be computed in polynomial time.

The reduction in the foregoing proof can also be used to show intractability for MinSuff, but only if we are able to guarantee that non-zero resource flow from $u \in Q$ to $c \in S$ does not occur. We illustrate this with an example: consider an X3C-instance with $q = 2$ and $|S| = 3$; the elements of $Q$ are indexed from 1 to 6 and $S = \{c_1, c_2, c_3\}$ with $c_1 = \{1, 2, 3\}$, $c_2 = \{1, 2, 4\}$ and $c_3 = \{1, 5, 6\}$; the answer for this instance is ‘no’ (one reason for this is that object 1 appears in all three sets $c_i$). Nevertheless, a feasible flow $f$ with number of minimal arcs equal to 1 exists, although $|C(f)| = 2$; one such flow is illustrated in Figure 2. The complexity status of problem MinSuff remains open. A similar reduction for the weighted version of MinSuff seems to be straightforward; other generalizations are studied in [25].

For MaxIncomp, we have the following result:

**Theorem 4.** Problem MaxIncomp is NP-hard.

**Proof:** It was noted before that minimization of $|T(A \cup X)|$ (by varying the ES-selection $X$) is equivalent to minimization of $|T(A \cup C(f))|$ (by varying the resource flow $f$). We use these two quantities interchangeably in the description that follows, and we will focus on minimizing $\text{obj} = |T(A \cup C(f)) \setminus A|$.

For an arbitrary instance of X3C, we construct an RCPSP-instance as follows. $N$ contains $\{0, z, t\} \cup Q \cup S$, where 0 and $t$ are dummy start and end, respectively (and thereby also predecessor or successor respectively, of all other activities), and $z$ is an extra ‘enforcer’ activity. Furthermore, for all $c \in S, u \in Q : (c, u) \in A \iff u \in c$. We have one resource type ($|K| = 1$) with availability $a_1 = 3|S|$, with resource usage $r_c = 3$ for each $c \in S$ and $r_u = 1$ for all $u \in Q$; $r_z = 3|S| - |Q|$. We define $LB = |S| - q$ and consider $(3q + 1)$
Figure 2: A feasible resource flow $f$ with the number of minimal arcs equal to 1, namely $(4, c_3)$, although $C(f) = \{(c_2, c_3), (4, c_3)\}$. The graph shows all arcs in $\phi(f)$ except those incident to 0 and $t$; flow values are indicated on the arcs.

additional sets of $LB$ activities, denoted as sets $E_i$ with $i = 0, 1, \ldots, 3q$, and we construct an arbitrary bijection $\pi : Q \mapsto \{1, 2, \ldots, 3q\}$. We include each of the sets $E_i$ into $N$. We also add $(e, u)$ to $A$ for each $e \in E_0, u \in Q \cup \{z\}$, as well as $(u, e)$ for each $u \in Q, e \in E_{\pi(u)}$. We set $r_e = 0$ for each $e \in E_i, i = 0, 1, \ldots, 3q$. Although the resulting RCPSP-instance is large, it can be constructed in time polynomial in the size of the X3C-instance.

Any feasible resource flow $f$ that routes resource units from or to $Q$-jobs via arcs not in $A$ immediately brings $obj$ above $LB$. We also see that any $c_1 \in S$ cannot acquire its resource units from another activity $c_2 \in S$ or from $z$ without raising $obj$ above $LB$. This is true as long as there are no two identical elements in $S$, and this property can easily be made to hold for the input instance of X3C by removing all but one of such duplicate sets: the ‘yes’ and ‘no’ answers remain the same before and after such removal. Consequently, any flow $f$ with $obj \leq LB$ necessarily has $f(0, c, 1) = 3$ for each $c \in S$. The value $LB$ constitutes a lower bound on $obj$, since for any flow $f$, $z$ will obtain its resource units from at least $LB$ different elements of set $S$.

We claim that the answer to X3C is ‘yes’ if and only if $obj = LB$. Suppose that the answer to X3C is ‘yes’: $q$ elements of $S$ can then each be associated with the three elements of $Q$ it contains, and send the resources they use to these $Q$-activities without cost (because each of these links is already in $A$). For each of the $LB$ remaining elements $c$ of $S$, one arc $(c, z)$ can be added, which results in $obj = LB$. 

9
Suppose now that the answer to X3C is ‘no’: we will then need more than \( LB \) additional flow-carrying arcs to supply \( z \) with its resource requirements, and so \( obj > LB \).

Consequently, the answer to any X3C-instance is ‘yes’ if and only if \( obj = LB \). This concludes the proof for MaxIncomp.

\[ \square \]

## 5 Resource allocation for an input schedule

Practical scheduling problems, apart from having multiple objectives, are also complex due to the number and variety of the constraints themselves, many of which are ‘soft’ constraints rather than ‘hard’ technological constraints, e.g. potentially relaxable human preference constraints. In practice, the schedule will often be the result of a ‘satisficing’ pass with respect to these multiple objectives and constraints [11, 19]. The schedule may need to combine the work-organization policies of the organization performing the project with the outcome of negotiations between the organization and its suppliers, its subcontractors and its customers. This necessity of producing a negotiated schedule before the start of the project’s execution that is in line with all parties involved (clients and suppliers, as well as workers and other resources) is especially present in multi-project environments (see De Boer [8], Hans et al. [15] and Leus and Herroelen [27] for a discussion of hierarchical planning frameworks for project-based organizations). The major functions of such a predictive schedule or baseline schedule are to serve as a basis for resource allocation and to be the starting point for planning external activities such as material procurement, preventive maintenance and shipment timing projections towards customers (see, for instance, Aytug et al. [4], Mehta and Uzsoy [29] and Wu et al. [40]).

The baseline schedule will be a guideline during project execution for all parties engaged. Uncertainties during the execution give rise to what is called dynamic or reactive scheduling or re-scheduling; ES-policies constitute an option for re-scheduling when the project is ongoing. For a given feasible schedule \( s \), we define the schedule-induced order \( R(s) = \{(i, j) \in N \times N|i \neq j \land s_i + d_i \leq s_j\} \), which corresponds to the precedence constraints implied by \( s \). Obviously, the schedule \( s \) needs to respect the technological precedence constraints inherent in \( A \) as well as the resource constraints. \( R(s) \) is an interval order, and has been introduced and studied in [31, 36]. For a feasible schedule \( s \), we say that a feasible resource flow \( f \) is compatible with \( s \) if \( \phi(f) \subseteq R(s) \). For instance, both the flows \( f_1 \) and \( f_2 \) of Figure 3 of the companion paper [26] are compatible with the schedule shown in Figure 2(a) in the same reference (with \( s_0 = 0 \) and \( s_5 = 3 \)). A compatible \( f \) represents a
detailed resource-allocation decision for schedule $s$. In the context of dynamic scheduling, the adoption of an ES-policy defined by a resource flow that is compatible with the baseline schedule, is a possibility for dynamic schedule ‘repair’ in case the actual activity durations are different from those in the baseline. The following lemma from [27] shows that this latter option is always a feasible alternative.

**Lemma 1.** For every feasible schedule there is at least one compatible feasible flow.

We define the following related optimization problems (the first letter ‘S’ in each name refers to the fact that a feasible schedule $s$ is part of the input, as opposed to the problems studied in Section 4): SMinFlowArcs, which is the same as its counterpart MinFlowArcs apart from the fact that we impose $\phi(f) \subset R(s)$, and SMinSuff and SMaxIncomp, which are the same as MinSuff and MaxIncomp apart from the fact that we require $X \subset R(s)$. Note that the new problems are not sub-problems of their counterparts without schedule since we restrict the set of solutions and not the input parameters, and that $G(N, R(s))$ is acyclic. We have the following result:

**Theorem 5.** Problems SMinSuff, SMinFlowArcs and SMaxIncomp are NP-hard.

**Proof:** For an arbitrary instance of X3C we construct an RCPSP-instance as follows. $N = \{0, z, t\} \cup Q \cup S$, with $z$ an extra ‘enforcer’ activity. 0 and $t$ represent the dummy start and end activity, respectively. As for the precedence constraints, for all $c \in S, u \in Q$: $(c, u) \in A \iff u \in c$. There is only one resource type, with resource usage $r_c = 3$ for $c \in S$, $r_u = 1$ for $u \in Q$ and $r_z = 3|S| - |Q|$; availability $a_1 = 3|S|$. All activities in $N \setminus \{0, t\}$ have unit durations. We construct a feasible schedule $s$ for this RCPSP-instance by assigning start times $s_c = 0$ ($c \in S$), $s_z = 1$, $s_u = 1$ ($u \in Q$), and $s_t = 2$. The schedule and resource usage are illustrated in Figure 3.

For the minimization of $|C(f)|$, the reasoning is completely similar to the way in which we associated the value of quantity $obj$ to the answer to X3C in the proof of Theorem 4.

Since the arcs used to route the resource units in this proof are all minimal for the resulting resource flow, and since there are no opportunities for lowering the number of minimal arcs, the X3C-instance can also be solved by constructing a flow with a minimum number of minimal arcs, so by using an algorithm for solving problem SMinSuff as a subroutine.

Finally, the theorem is also valid for SMaxIncomp because each of the arcs added in the selection adds only the minimum possible number of arcs (namely one) to the transitive closure of the resulting extended network. □
Another set of problems of interest to us is based on ‘float’ or ‘slack’ values. Policella [35] and Mehta and Uzsoy [29], for instance, suggest using a measure of average ‘float’ in the baseline, relative to schedule length. Depending on the actual objective function, various float quantities can be used (cf. [39]). Al-Fawzan and Haouari [2], Lambrechts et al. [22] and Schwindt [37] propose the sum of free floats as a robustness measure for a schedule while Braeckmans et al. [6], Hazir [16], Hazir et al. [17], Liu et al. [28] and Leon et al. [24] investigate total floats. The free-float objective is of particular interest when ‘stability’ or ‘predictability’ is an issue, that is when management wants the actually executed schedule not to deviate much from the initial baseline schedule (see, for instance, [18, 29]). An extensive review of schedule robustness measures is provided in Ghezaiel et al. [13].

We define the pairwise float between activities $i$ and $j$ to be $\delta_{ij} = s_j - s_i - d_i$, and the minimum sum of pairwise floats $\Delta_{ij}(E)$ as the minimum sum of values $\delta_{kl}$, summed over all edges $(k, l)$ on a path from $i$ to $j$ in graph $G(N, E)$, with $E$ an arbitrary order relation on $N$, minimized over the possibly multiple paths; $\Delta_{ij}(E)$ is defined only for activity pairs $(i, j) \in E$.

We investigate the following problems:

Problem SMaxSumFF

**Instance:** RCPSP-instance $\Gamma(N, A, d, r, a)$, feasible schedule $s$, integer weights $w_i$ for $i \in N \setminus \{n\}$.

**Goal:** Find a sufficient set $X \subset R(s)$ that maximizes $\sum_{i \in N \setminus \{n\}} w_i \min_{(i,j) \in T(A \cup X)} \delta_{ij}$. 

Figure 3: The schedule and resource usage for the proof of Theorem 5.
The allowable increase in the duration of an activity without effect on the starting times of its successors is called ‘free float’ or ‘FF’ for short, which explains the choice of the name for this problem.

Problem SMaxSumTF

Instance: RCPSP-instance \( \Gamma(N, A, d, r, a) \), feasible schedule \( s \), integer weights \( w_i \) for \( i \in N \setminus \{n\} \).

Goal: Find a sufficient set \( X \subset R(s) \) that maximizes \( \sum_{i \in N \setminus \{n\}} w_i \Delta_{in}(T(A \cup X)) \).

The protection of \( s_n \) from delays in the completion of activity \( i \) is called ‘total float’ or ‘TF’ of \( i \), whence the name of this last problem. As an illustration, for the schedule in Figure 2(a) of [26] (with \( s_5 = 3 \)) and with sufficient set \( C(f_1) \) resulting from flow \( f_1 \) as described by Figure 3(a) in [26], we have free-float values of 0, 1, 0, 0, 0 and total float equal to 0, 1, 0, 0, 1 for activities 0, 1, 2, 3 and 4, respectively.

In the results below, we refer to the following decision problem:

Problem 3-PARTITION

Instance: A finite set \( Q \) of 3\( q \) elements, a positive integer \( B \), and an integer size \( u(i) > 0 \) for each \( i \in Q \), such that each \( u(i) \) satisfies \( B/4 < u(i) < B/2 \) and such that \( \sum_{i \in Q} u(i) = qB \).

Question: Can \( Q \) be partitioned into \( q \) disjoint subsets \( Q_1, Q_2, \ldots, Q_q \) such that, for \( 1 \leq i \leq q \), \( \sum_{j \in Q_i} u(j) = B \)?

3-PARTITION is known to be NP-complete (problem SP15 in [12]).

Theorem 6. Problem SMaxSumTF is NP-hard, even with weights \( w_i \in \{0, 1\} \).

Proof: For an arbitrary instance of 3-PARTITION, we construct an instance of RCPSP as follows. The set of activities to be scheduled is \( N = \{0, z, t\} \cup Q \). Activities 0 and \( t \) are dummy start and end, respectively. Apart from these dummy precedence constraints, \( A \) is empty. There is a single resource type \((|K| = 1)\), with resource usage \( r_z = q \) and \( r_i = 1, i \in Q \); the availability \( a_1 = q \). We set \( d_z = 1 \) and \( d_i = u(i) \) for each \( i \in Q \). We construct an arbitrary bijection \( \pi : \{1, 2, \ldots, 3q\} \mapsto Q \) and produce a feasible schedule \( s \) for this RCPSP-instance by assigning starting times \( s_z = 0, s_{\pi(1)} = 1 \) and \( s_{\pi(i)} = s_{\pi(i-1)} + d_{\pi(i-1)} \) for \( i = 2, 3, \ldots, 3q; s_t = qB + 1 \). The schedule and resource usage are illustrated in Figure 4. We set weight \( w_z = 1 \); all other weights are zero.

We claim that the answer to the 3-PARTITION-instance is ‘yes’ if and only if the optimal objective-function value for the SMaxSumTF-instance is \((q - 1)B\). In that case, each of the \( q \) resource units is assigned to (three)
elements of $Q$ that sum to exactly $B$, and the partition of $Q$ can follow this resource allocation. Verification of this claim is straightforward. \hfill \square

**Theorem 7.** Problem SMaxSumFF is NP-hard.

**Proof:** For an arbitrary instance of 3-PARTITION, we construct an instance of RCPSP as follows. The set of activities to be scheduled is $N = \{0, t\} \cup Q \cup \{z_1, z_2, \ldots, z_q\}$, with $0$ and $t$ the dummy start and end activities. Apart from the dummy precedence constraints, $A$ is empty. There is a single resource type ($|K| = 1$), with resource usage $r_i = u(i), i \in Q$, and $r_{z_i} = iB, i = 1, \ldots, q$; the availability $a_1 = qB$. All activities in $N \setminus \{0, t\}$ have unit durations. We construct a feasible schedule $s$ for this RCPSP-instance by assigning starting times $s_i = 0$ for $i \in Q$, $s_{z_i} = i$ for $i = 1, \ldots, q$ and $s_t = q+1$. The schedule and resource usage are illustrated in Figure 5.

The weights are chosen in the following way: $w_0 = 0$, $w_{z_i} = 0$ for $i = 1, \ldots, q$, and $w_i = u(i)$ for $i \in Q$. We assert that the answer to the 3-PARTITION-instance is ‘yes’ if and only if the optimal objective-function value for the SMaxSumFF-instance is $Bq(q - 1)/2$. In that case, set $Q$ can be partitioned into (three-element) subsets with equal sums of values $u(i)$. Verification of the validity of this statement is not difficult. \hfill \square

This last result is somewhat less satisfactory than Theorem 6, since we allow general values for the weights.

The complexity results of Sections 4 and 5 confirm that resource allocation is usually difficult even when objective-function evaluation by itself is not intractable. They also indicate that even the use of ‘surrogate’ objective functions, which has been suggested by a number of recent sources, leads to hard problems, thus justifying the use of solution approaches by means of integer-programming techniques or ‘dedicated’ implicit enumeration.
6 Joint resource allocation and scheduling

Finally, we briefly discuss joint scheduling and resource allocation: this combined problem does not seem to be easier than resource allocation for a given schedule. Actually, this problem deserves study in its own right only if the schedule is an argument to the objective function, which is the case only for the float-based objectives. We define problems MaxSumTF and MaxSumFF (without initial letter ‘S’ in the names) similarly as their counterparts SMaxSumTF and SMaxSumFF, but a feasible schedule is now not part of the input but of the output. We also include a deadline $D$ on schedule length, since both total float as well as free float will tend to increase with increasing makespan. We show that both these new problems are difficult:

**Theorem 8.** Problems MaxSumFF and MaxSumTF are NP-hard, even with weights $w_i \in \{0, 1\}$.

**Proof:** For an arbitrary instance of 3-PARTITION, we construct an instance of RCPSP as follows. The set of activities to be scheduled is $N = \{0, z, t\} \cup Q$. $0$ and $t$ are dummy start and end, respectively. Apart from the precedence constraints involving $0$ and $t$, $A$ is empty. Let $W$ equal the sum of the three largest $u(i)$-values in the 3-PARTITION-instance. We choose $|K| = 1$ (one resource type) with resource usage $r_z = q$ and $r_i = 1, i \in Q$; the availability $a_1 = q$. Activity $z$ has unit duration, and for $i \in Q$: $d_i = u(i)$. Weight $w_z = 1$; all other weights are zero. The deadline $D$ is chosen as $W + 1$.

Given our choice of $W$, a feasible schedule always exists. We claim that the answer to 3-PARTITION is ‘yes’ if and only if the total float or free
float of \( z \) is \( W - B \). For the objective function (any of the two) to take on this value, the resource profile of the constructed schedule needs to contain a ‘gap’ in the utilization of each individual resource unit, of length \( W - B \). Consequently, activities from \( Q \) can only occupy \( B \) or less time on each resource unit. If we take into account the duration of the \( Q \)-activities, we see that the objective function is \( W - B \) only if activities from \( Q \) occupy a time period of exactly \( B \) on each resource unit. The allocation of jobs from \( Q \) to resource units therefore corresponds with the desired partition of \( Q \).

We illustrate the foregoing reduction by means of an example. Consider the following instance of 3-PARTITION: \( Q \) contains 12 elements, indexed 1 to 12, with size \( u(i) = 3 \), \( i = 1, \ldots, 8 \) and \( u(i) = 4 \), \( i = 9, \ldots, 12 \). We have \( q = 4 \), \( B = 10 \) and \( W = 12 \), so we set \( a_1 = 4 \) and \( D = 13 \). Figure 6 shows a schedule that achieves total and free float of \( W - B = 2 \) for \( z \) and the answer to the 3-PARTITION-instance therefore is ‘yes’: multiple satisfying partitions exist, one being \( \{\{1, 5, 9\}, \{2, 6, 10\}, \{3, 7, 11\}, \{4, 8, 12\}\} \), which corresponds with the resource allocation in the figure.

7 Summary

In this article, we have studied the relationship between resource allocation and ES-policies, which are a type of scheduling policies for stochastic scheduling. We have established the intractability of scheduling with ‘surrogate’ objective functions that have recently been proposed by a number of researchers. Our results confirm that resource allocation is usually difficult even when objective-function evaluation by itself is not intractable, thus justifying solution approaches by means of implicit enumeration or integer-programming techniques. We have also emphasized the fact that for some of
these surrogate measures, all optimal solutions can be dominated for all regular objective functions, which underlines the importance of a well-considered choice for such alternative quality measures.

Complexity results have been obtained both with and without input schedule, including the cases where the objective is the sum of the free floats or total floats for a given schedule, as well as when the objective is a measure for the extra constraints imposed by the ES-policy on top of the (input) technological precedence constraints; all intractability results were obtained for a single resource type. The complexity status of some of the problems that were introduced remains open.

References


